## Mikroprogramme

Alle Vorgänge, die während eines Taktes (2 Phasen!) im Mikroprozessor ablaufen, werden durch die Angabe derjenigen Tore festgelegt, die dabei zu öffnen sind. Diese Angabe heißt Mikrobefehl. Jeder ***Maschinenbefehl*** wird durch ***einen oder mehrere Mikrobefehle*** realisiert.

Neben dem Arbeitsspeicher (RAM) steht in MIKROSIM auch ein Mikroprogrammspeicher (MPS) zur Verfügung, in dem ***Sequenzen von Mikrobefehlen***, also ganze **Mikroprogramme** abgelegt werden können. Nun kann das Steuerwerk selbst die in den einzelnen Mikrobefehlen angegebenen Tore zum Öffnen auswählen, ein Eingriff von Hand erübrigt sich. Vom Steuerwerk aus gehen hierzu Steuerleitungen zu jedem Tor. Diese Leitungen werden bei MIKROSIM der Übersichtlichkeit wegen nicht dargestellt.

Ein Mikrobefehl besitzt bei MIKROSIM 24 Bit. Die ersten 16 Bit sind von 0 bis F nummeriert und den gleichnamigen Toren zugeordnet (Steuerleitungen!). Sie lassen sich einzeln per Doppelklick oder durch Antippen der Leertaste manipulieren. Die letzten 8 Bit sind zu einer zweistelligen Hex-Zahl zusammengefasst und legen fest, bei welcher Platznummer die Ausführung des Mikroprogramms fortgesetzt wird (FA = Folgeadresse). Normalerweise ist FA um 1 größer als die aktuelle Platznummer (Voreinstellung), doch sind Abweichungen davon jederzeit möglich. So ist z.B. ***beim letzten Mikrobefehl einer zusammengehörenden Sequenz FA auf 00*** zu setzen, weil dies für MIKROSIM das Signal ist, die automatische Ausführung anzuhalten. Links vor der Platznummer eines jeden Mikrobefehls ist Platz für eine kurze Bemerkung. Vor dem ersten Befehl einer Sequenz trägt man hier ein Mnemonik für den zu realisierenden Maschinenbefehl ein.

*Beispiel:* ***AX = AX – 2 * BX***

| | Befehl | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | FA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | MOV DR, BX | | | | x | | | | | | | x | | | | | | 01 |
| 01 | SUB AX, DR | x | x | | | | | | | | x | | | | x | | | 02 |
| 02 | SUB AX, DR | x | x | | | | | | | | x | | | | x | | | 00 (!) |

Um Mikrobefehle einzugeben, wird mit der *rechten Maustaste* oder *Doppelklick* in den entsprechenden Bereich auf dem Bildschirm geklickt.
(*Einstellungen → Tor statt Kreuz* zur Darstellungsauswahl: Tornummer oder Kreuz)
Einzelne Mikrobefehle können mit *Strg Entf* gelöscht werden, Platz für neue Zeilen schafft man mit *Strg Einfg*. Die Folgeadressen werden dabei automatisch angepasst.

**Aufgabe 1**
Teste das Programm aus dem Beispiel (Torsteuerung → Automatisch aus MPS)

**Aufgabe 2**
Schreibe und teste die folgenden Mikroprogramme. Du musst für jedes Programm den alten Programmspeicher erstmal abspeichern und dann leeren.
**a)** AX = BX + 3 * AX
**b)** AX = [IP] *(MOV AX , [IP] : der Inhalt der durch IP angegebenen RAM-Adresse soll in AX transportiert werden. Benutze dazu das Adressregister AR)*
**c)** INC [AX] *(Der Inhalte des durch AX indizierten RAM-Platzes soll um 1 erhöht werden)*
**d)** DEC [IP] *(Der Inhalte des durch IP indizierten RAM-Platzes soll um 1 verringert werden)*
**e)** AX = [BX] + [BX +1] *(die Inhalte der RAM-Adressen [BX] und [BX+1] sollen addiert werden)*

Die Tabelleneingabe kann als Maschinensprache auch vereinfacht als Binärfolge angegeben werden. So wird noch einmal der Zusammenhang zwischen Hochsprache, Assembler und Maschinensprache deutlich

| | |
|---|---|
| Hochsprache | AX = AX - 2 * BX |
| Assembler | MOV DR, BX<br>SUB AX, DR<br>SUB AX, DR |
| Maschinensprache (Mikrobefehle) | 0001 0000 0010 0000 (Tore 3, A → Takt)<br>1100 0000 0100 0100 (Tore 0,1, 9, D → Takt)<br>1100 0000 0100 0100 (Tore 0,1, 9, D → Takt) |

Der Mikrorechner ist nun schon in der Lage, zusammenhängende Mikrobefehle abzuarbeiten. Da in der Regel stets nur ein Mikroprogramm-Speicher zur Verfügung steht, ist unser Rechner noch sehr begrenzt. Um ihn flexibel zu machen, müssen die einzelnen Mikrobefehle und deren zeitlicher Aufruf getrennt werden.

## Hol- und Ausführungsphase

**Aufgabe 3**
**a)** Ergänze folgenden Mikrospeicherinhalt in den Zeilen 00 - 06:

| NR | Befehl | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | FA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | MOV OP, [IP] | | | | | | | | | | | | | | | | | 01 |
| 01 | | | | | | | | | | | | | | | | | | 02 |
| 02 | | | | | | | | | | | | | | | | | | 03 |
| 03 | INC IP | | | | | | | | | | | | | | | | | 04 |
| 04 | MOV DR, [IP] | | | | | | | | | | | | | | | | | 05 |
| 05 | | | | | | | | | | | | | | | | | | 06 |
| 06 | INC IP | | | | | | | | | | | | | | | | | 07 |
| 07 | do command | | | | | | | | | | | | | | | | **X** | 00 |
| 08 | ---------- | | | | | | | | | | | | | | | | | 09 |
| 09 | LOAD AX, nn | 0 | | | | | | | | | 9 | | | | | | | 00 |
| 0A | LOAD BX, nn | | | 2 | | | | | | | 9 | | | | | | | 00 |
| 0B | ---------- | | | | | | | | | | | | | | | | | 0C |
| 0C | ADD BX, AX | | | | 3 | | | | | | | A | | | | | | 0D |
| 0D | | | 1 | 2 | | | | | | | 9 | | | | | | | 00 |
| 0E | | | | | | | | | | | | | | | | | | 0F |
| 0F | | | | | | | | | | | | | | | | | | 10 |

**b)** Belege die ersten vier Zellen des RAM mit den Werten 09, 05, 0A, und 03 wie im Bild.

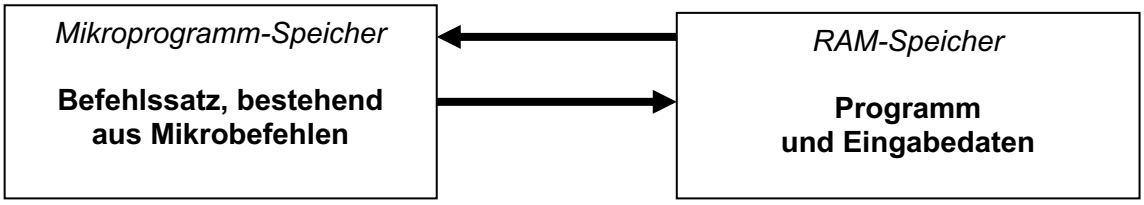
| *Arbeitsspeicher (RAM)* | | | | | |
|---|---|---|---|---|---|
| RAD | 00 | 01 | 02 | 03 | 04 |
| 00 | **09** | **05** | **0A** | **03** | 00 |
| 01 | 00 | 00 | 00 | 00 | 00 |

Führe das Programm **schrittweise (1 Takt)** aus und verfolge den Weg, den die RAM-Belegungen 09, 05, 0A und 03 zurücklegen. Vor dem Start auf „Register löschen“ klicken.

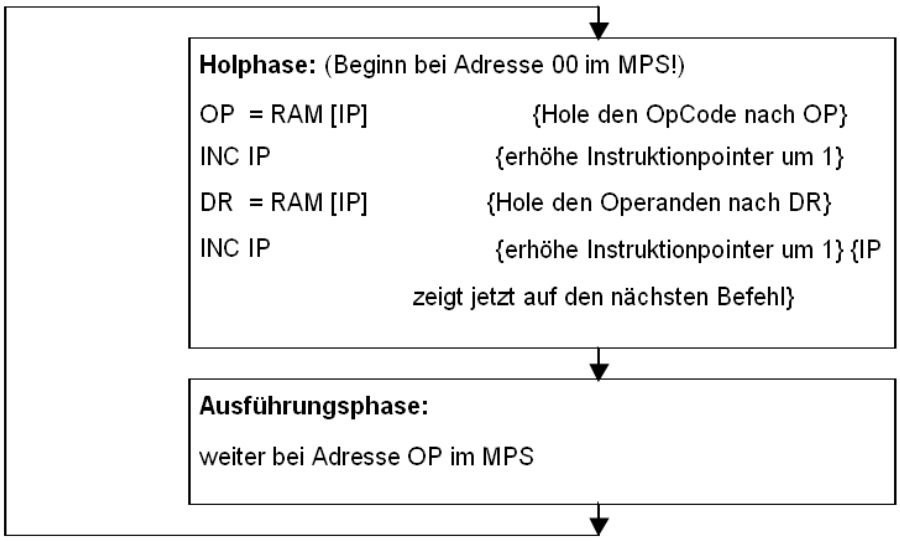
**c)** Was macht das Programm?

Bisher wurde so gearbeitet, die Folgeadresse eines Befehls stets gleich dem Inhalt von FA war. Das ist allerdings nur der Fall, wenn das Tor F (wie bisher immer) geschlossen ist. In Wirklichkeit wird die Folgeadresse nämlich durch ein nicht dargestelltes Addierwerk als **Summe von FA und dem durch *Tor F* gelieferten Wert** berechnet. Ist ***Tor F*** geschlossen, so liefert es den Wert 00, ist es geöffnet, so liefert es den Inhalt von OP. Tor F kann nicht von Hand, sondern nur per Mikrobefehl geöffnet werden.

Damit ein Maschinenprogramm vollautomatisch ausgeführt werden kann, werden ***alle seine Befehle komplett im RAM gespeichert*** sein (bisher standen dort nur die Daten). Die folgenreiche Grundidee (*ohne die eine Computergeneration wie die heutige nicht möglich geworden wäre*) einer automatischen Ausführung eines ***universellen Programms*** ist somit die Trennung vom Befehlssatz und dem Programm mit Eingabedaten.

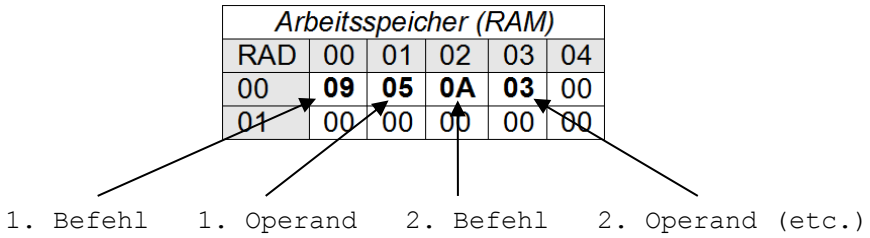



Hierzu ist es notwendig, für jeden Maschinenbefehl sowohl die Operation als auch den Operanden (falls vorhanden) durch eine zweistellige Hex-Zahl zu codieren. Als Operationscode bietet sich die Startadresse des zugehörigen Mikroprogramms im MPS an, der Operand ist ohnehin eine Zahl (je nach Adressierungsart verschieden interpretiert).



Im Normalfall stellen wir jeden Maschinenbefehl im RAM also durch zwei aufeinander folgende Bytes dar.

Bevor ein Maschinenbefehl ausgeführt werden kann, muss das **erste** dieser beiden Bytes nach **OP**, das **zweite** nach **DR** gebracht werden **(Holphase).** Anschließend kann dann die Ausführung der für diesen Befehl zuständigen Mikroprogramme angestoßen werden **(Ausführungsphase).** Ist die Ausführung beendet, so muss sich die Holphase für den nächsten Maschinenbefehl anschließen. Damit der Mikroprozessor ihn findet, muss die RAM-Adresse dieses Befehls in einem Register zur Verfügung stehen. Wir benutzen IP als Befehlszeiger (engl. Instruction-Pointer).




Um nun universell programmieren zu können, muss der Mikroprogrammspeicher erweitert werden.

**Aufgabe 4**
Vervollständige den Mikroprogrammspeicher **(nächste Seite)** um die angegebenen Assembler-Befehle und speichere den Inhalt. Mit dem Mikroprogrammspeicher wird von nun an programmiert.
**„nn“** bezeichnet die Zahl, die durch den **Operanden** angegeben ist. **Nach der Holphase** steht der **Operand im Register DR**, daher ist könnte z.B. *LOAD AX, nn* auch *LOAD AX, DR* geschrieben werden (soll es hier aber nicht).

## Assembler Darstellung

Schalte in Mikrosim die **Assembler Darstellung** unter „RAM-Anzeige“ an. Von nun an wird die Programmierung komfortabler.
Hinweis: An der Stelle nn wird der konkrete Wert eingegeben, Mikrosim übernimmt den Operanden dann automatisch.
Bsp: LOAD AX, 05  oder ADD BX, [0A]

**Aufgabe 5**
**a)** Schreibe mit den obigen Mikrobefehlen ein Programm, das die beiden Zahlen *0F* und *0A* (1. und 2. Operand) subtrahiert und das Ergebnis in AX abspeichert.

**b)** Schreibe ein Programm, das folgende Berechnung der Hexadezimalzahlen ausführt
*05 + 0F + 0A – 0B - 06 - 09*
und das Ergebnis in AX abspeichert (Ergebnis in AX müsste 04 sein).

**c)** Schreibe ein Programm, das die folgende Befehlskette ausführt:
[1C] = [1D] + [1E] + [1F]

| NR | Befehl | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | FA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **00** | MOV OP, [IP] | | | | | | 5 | | | 8 | | | | | | | | **01** |
| **01** | | | | | | | | | | | | | | C | | | | **02** |
| **02** | | | | | | | | 6 | | | 9 | | | | | | | **03** |
| **03** | INC IP | | | | | 4 | 5 | | 7 | | | | | | | | | **04** |
| **04** | MOV DR, [IP] | | | | | | 5 | | | 8 | | | | | | | | **05** |
| **05** | | | | | | | | | | | | | | C | | | | **06** |
| **06** | INC IP | | | | | 4 | 5 | | 7 | | | | | | | | | **07** |
| **07** | do command | | | | | | | | | | | | | | | | **F** | **00** |
| **08** | ---------- | | | | | | | | | | | | | | | | | |
| **09** | LOAD AX, nn | | | | | | | | | | | | | | | | | |
| **0A** | LOAD BX, nn | | | | | | | | | | | | | | | | | |
| **0B** | ---------- | | | | | | | | | | | | | | | | | |
| **0C** | ADD BX, AX | | | | | | | | | | | | | | | | | |
| **0D** | | | | | | | | | | | | | | | | | | |
| **0E** | ADD BX, [nn] | | | | | | | | | | | | | | | | | |
| **0F** | | | | | | | | | | | | | | | | | | |
| **10** | | | | | | | | | | | | | | | | | | |
| **11** | ADD AX, [nn] | | | | | | | | | | | | | | | | | |
| **12** | | | | | | | | | | | | | | | | | | |
| **13** | | | | | | | | | | | | | | | | | | |
| **14** | ---------- | | | | | | | | | | | | | | | | | |
| **15** | SUB BX, AX | | | | | | | | | | | | | | | | | |
| **16** | | | | | | | | | | | | | | | | | | |
| **17** | SUB BX, [nn] | | | | | | | | | | | | | | | | | |
| **18** | | | | | | | | | | | | | | | | | | |
| **19** | | | | | | | | | | | | | | | | | | |
| **1A** | SUB AX, [nn] | | | | | | | | | | | | | | | | | |
| **1B** | | | | | | | | | | | | | | | | | | |
| **1C** | | | | | | | | | | | | | | | | | | |
| **1D** | ---------- | | | | | | | | | | | | | | | | | |
| **1E** | MOV BX, [nn] | | | | | | | | | | | | | | | | | |
| **1F** | | | | | | | | | | | | | | | | | | |
| **20** | | | | | | | | | | | | | | | | | | |
| **21** | MOV [nn], BX | | | | | | | | | | | | | | | | | |
| **22** | | | | | | | | | | | | | | | | | | |
| **23** | | | | | | | | | | | | | | | | | | |
| **24** | MOV AX, [nn] | | | | | | | | | | | | | | | | | |
| **25** | | | | | | | | | | | | | | | | | | |
| **26** | | | | | | | | | | | | | | | | | | |
| **27** | MOV [nn], AX | | | | | | | | | | | | | | | | | |
| **28** | | | | | | | | | | | | | | | | | | |
| **29** | | | | | | | | | | | | | | | | | | |
| **2A** | ---------- | | | | | | | | | | | | | | | | | |
| **2B** | INC AX | | | | | | | | | | | | | | | | | |
| **2C** | DEC AX | | | | | | | | | | | | | | | | | |
| **2D** | INC BX | | | | | | | | | | | | | | | | | |
| **2E** | DEC BX | | | | | | | | | | | | | | | | | |
| **2F** | ---------- | | | | | | | | | | | | | | | | | |
| **30** | HALT | | | | | 4 | | | 7 | | | | | | | | | **40** |
| **31** | ---------- | | | | | | | | | | | | | | | | | |

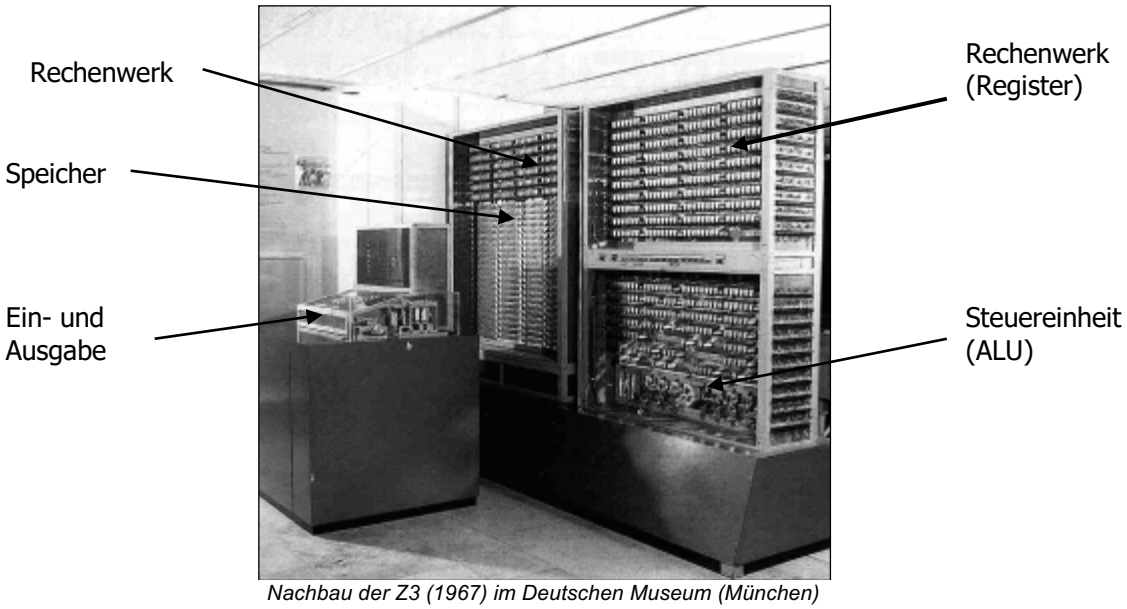
***Exkurs***

*Die **Z3** – 1941 von Konrad Zuse gebaut – war die erste frei programmierbare, auf dem binären Zahlensystem basierende Rechenmaschine der Welt. Sie wird daher heute, vor allem im deutschsprachigen Raum, oft als erster funktionsfähiger programmierbarer Rechner bezeichnet. 1944 wurde die Z3 durch einen Bombenangriff zerstört.*

*Die Z3 ist eine getaktete Maschine. Die Taktung wird von einem Elektromotor übernommen, der eine so genannte Taktwalze antreibt. Diese ist eine Trommel, welche sich ca. 5.3 mal pro Sekunde dreht, und während einer Drehung die Steuerung der einzelnen Relaisgruppen übernimmt. Die Z3 verfügt über folgende Maschinenbefehle:*

- *Pr z Speicherzelle z in Register R1/R2 laden*
- *Ps z R1 in Speicherzelle z schreiben*
- *Ls1 Addition R1 := R1 + R2*
- *Ls2 Subtraktion R1 := R1 - R2*
- *Lm Multiplikation R1 := R1 * R2*
- *Li Division R1 := R1 / R2*
- *Lw Quadratwurzel R1 := SQRT(R1)*
- *Lu Dezimalzahl einlesen in R1 / R2*
- *Ld R1 als Binärzahl ausgeben*

*Über die Tastatur können alle Operationen außer den Speicherzugriffen (Pr und Ps) direkt ausgeführt werden. Befehle werden über Lochstreifen eingelesen, sie stehen noch nicht im Speicher. Jeder Befehl auf dem Lochstreifen wird mit 8 Bit kodiert. Die Z3 kennt keine Sprungbefehle.*




*Nachbau der Z3 (1967) im Deutschen Museum (München)*